TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie
Studijní obor: Informační technologie

Textové rozhraní pro nástroj YUM

Text interface for YUM tool

Bakalářská práce

Autor: Nikola Čulík
Vedoucí práce: Mgr. Milan Keršláger

V Liberci 17. 5. 2013

Prohlášení
Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 21/2000 Sb. o právu autorském, zejména § 60 – školní dílo. Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL. Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případěmá TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skute čné výše.

Diplomovou práci jsem vypracoval(a) samostatněs použitím uvedené literatury a na základěkonzultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

Abstrakt

Uživatelské rozhraní tvoří důležitou součást každého programu, tvoří pro nezkušeného uživatele ovádací prvky kterými potom může ovládat složitější systémy mnohem jednodušeji. Vzhledem k možné náročnosti systému je kladen důraz na pochopitelnost rozhraní a jeho jednoduchost, ale zároveň na nenáročnost potřebnou k tomu aby systém mohl pracovat výkonně a nebyl rozhraním brzděn. Textové rozhraní je proto vhodným prostředkem k docílení zmiňovaných vlastností. Nástroj YUM v linuxu zajišťuje instalaci a odinstalaci součástí systému pomocí příkazů nebo právě uživatelských rozhraní. Textovým uživatelským rozhraním získáme vhodnou pomůcku pro případ že uživatel ztratí grafické prostředí a je nucen pracovat v konzoli systému nebo pokud chce omezit zátěž systému během instalace na naprosté minimum. Program je vytvořen v programovacím jazyce Python neboť je součástí standartní distribuce Fedora a tedy není potřebné instalovat cokoliv k běhu programu na novém systému.

Abstract

The user interface is an important part of any program, make for an inexperienced user controls which can then control the complex systems much easier. Due to the potential performance of the system, emphasis is placed on comprehensible interface and its simplicity, but the modesty needed for the system to work efficiently and not be hindered by the interface. Text input interface is thus an appropriate means to achieve the above mentioned characteristics. YUM tool in Linux provides installation and uninstallation part of the system using commands or just the user interface. Text user interface can be used by user who lost advanced graphic interface, or if he wants to reduce system load during installation to an absolute minimum. The program is created in the programming language Python as a standard part of the Fedora distribution and therefore there is no need to install anything to run on the new system.

Table of Contents

1 Balíčkovací systém Linuxu (linux package service)

Balíčkovací systém, nebo jinak také nazývaný manažer balíčků, je skladba softwarových nástrojů na instalaci, upgrady, nastavení a odstraňování softwarového vybavení počítače a zároveň zachování konzistence. Většinou obsahuje databázi závyslostí a informace o verzích aby nedocházelo k chybám a přehmatům. Balíčky jsou distribuce softwaru, aplikací a dat. Obsahují také metadata jako název softwaru, jeho popis, účel, verzi a závyslosti jiných programů potřebných k jeho běhu. Po instalaci se metadata ukládají do lokální databáze. Balíčkovací systém je vytvořen aby šetřil čas, peníze a organizaci pomocí vzdálené správy a distribuce softwaru. Prakticky eliminuje nutnost manuální instalace a updatů. To může být velmi užitečné zváště pro velké korporace které používají Linux nebo jiný systém na základě Unixu, ty většinou používají tisíce nebo statisíce různých softwarových balíčků takže se manažer balíčků stal z prvotního ulehčení práce nutností. Balíčkovací systém pomáhá sestavovat zcela nové distribuce operačního systému zaměřené na specifickou funkci, například obsluha síťového serveru nebo zálohovacího zařízení(Clonezilla).

Správa balíčků je nejvýraznějším rysem každé linuxové distribuce. Současný trend u většiny velkých projektů nabízí možnost balíček vybrat a nainstalovat jí jako ve windows, například Synaptic v Debian Linuxu a nebo Drakrpm v Mandriva Linux. Trento typ je pouze jinak graficky zpracovaným Front-endem který obsluhuje balíčkovací systém, jde tedy pouze o jinak zpracované grafické prostředí které navozuje uživateli známý pocit narozdíl od učení se ovádat celý balíčkovací systém příkazy v konzole systému. Konzolový přístup má ovšem oproti grafickým aplikacím dvě velké výhody a to je rychlost a výkon.

Velkou nevýhodou je různorodost balíčkovacích systémů. Při změně distribuce je nutné se všechny, nebo většinu, příkazů pro ovládání přeučit a i během vývoje mnohdy dochází ke změnám příkazů. Pro orientaci v tomto složitém prostředí je možné použít dokumentace jednotlivých projektů ale i fanouškovské stránky na internetu, například *distrowatch.com*

Manažer který je vyvíjen největšími linuxovými prjekty se jmenuje rpm neboli Red Hat package manager, na jeho základu je postaveno několik manažerů pro tyto největší distribuce Jmenovitě **zypp** pro openSUSE, **yum** pro Fedoru a CentOS a **urpmi** pro distribuce Mandriva a Mageia.

Další významnou větví manažerů jsou navrženy pro systémy Linux odvozené z distribuce Slackware. Jak poznamenal při několika příležitostech Patrick Volkerding, zakladatel distribuce Slackware, je nepravděpodobné že by slackware měl někdy pokročilý balíčkovací manažer, tedy řešící závyslosti jednotlivých balíčků na sebe. V této distribuci jsou všechny úkony s balíčky prováděny kolekcí skriptů zvaných **pkgtools.** V současnosti používá Slackware distribuce pokročilejší **slackpkg**. Derivace Slackware linuxu využívají další své verze pokročilějších manažerů

Existují i manažery nezávyslé na cílovém systému které zaznamenaly různou míru úspěchu, například manažer Smart vyvinutý distribucí Conectiva využívají někteří uživatelé Mandriva linux raději než urpmi nebo zypperu v případě uživatelů openSUSE. Za zmínku rozhodně stojí packageKit, nativní pro distribuci Fedora, který se začíná rozmáhat jako manažer balíčků pro stále více distribucí.

Hlavní funkce

- Kontrola souborů aby byla zajištěna jejich správnost a kompletnost
- Kontrola digitálních podpisů by byl ověřen původ
- Upgrade softwaru na aktuílní verzi, většinou přes repozitář
- Seskupování balíčků dle funkce a účelu k omezení zmatení uživatele
- Správa závyslostí, aby byly staženy všechny potřebné balíčky které jsou potřeba k funkci právě instalovaného balíčku

2 YUM

YUM neboli Yellowdog Updater, Modified je open-source manažer balíčků pro Linuxové distribuce kompatibilní s RPM (Red Hat Package Manager). Byl vyvinut Sethem Vidalem a jeho skupinou dobrovolných vývojářů. YUM má vlastní interface v příkazovém řádku ale několik dalších nástrojů dodává YUMu grafické uživatelské rozhraní GUI.

YUM je kompletním přepisem jeho předchůdce Yellowdog Updateru (YUP) a proto byl primárně vyvíjen pro Red Hat linux systémy. Nyní se používá v mnoha linuxových distribucích jako Fedora,CentOS a Red Hat.

V mém projektu je YUM základním Back-endem.

Yum načítá svoji konfiguraci z jednoho nebo více souborů s nastavením. Hlavní konfigurační soubor se nalézá v adresáři /etc/yum.conf a obsahuje základní nastavení pro použití při získávání a instalaci balíčků. Jeho konfigurace může například vypadat takto:

```
$ cat /etc/yum.conf
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log
pkgpolicy=newest
distroverpkg=centos-release
tolerant=1
exactarch=1
retries=20
obsoletes=1
gpgcheck=1

# PUT YOUR REPOS HERE OR IN separate files named file.repo
# in /etc/yum.repos.d
```

Mimo tento soubor yum také hledá vzdálené repozitáře z souborů definicí umístěných v adresáři /ect/yum.repos.d. Definice repozitáře může vypadat takto:

```
[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$ba
search&repo=os
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-centos4
```

Každá definice začíná popisným jménem v závorkách. V každé definici je řádek se jménem, indikátorem zda se mají ověřovat GPG podpisy a umístěním balíčků a GPG klíčů použitých k jejich podpisu. Proměnné můžeme použít i v URL adrese jako v našem příkladu `$releasever` je proměnná simbolizující operační systém a `$basearch` umístění aktualizací.

Před updatem balíčků v systému je dobré zjistit jaké balíčky jsou již v systému nainstalovány. YUM obsahuje příkaz pro zobrazení nainstalovaných balíčků ve formě seznamu. Pokus je tento příkaz volán bez argumentů tak se v konzole vypíše například následující text:

```
$ yum list |more

Setting up Repos
Reading repository metadata in from local files
Installed Packages
4Suite.i386                             1.0-3
installed
GConf2.i386                             2.8.1-1
installed
GConf2-devel.i386                       2.8.1-1
installed
HelixPlayer.i386                        1:1.0.6-0.fc3.1
installed
ImageMagick.i386                        6.2.2.0-2.fc3
installed
MAKEDEV.i386                            3.13-1
installed
Maelstrom.i386                          3.0.6-6
installed
NetworkManager.i386                     0.3.4-1.1.0.fc3
installed
NetworkManager-gnome.i386               0.3.4-1.1.0.fc3
installed
```

Pokud se příkazu zadá jako argument string potom YUM vyhledá pouze balíčky se shodným jménem jako zadaný string. Zde je příklad výstupu po zadání argumentu „curl“:

```
  $ yum list curl

Setting up repositories
rawhide-updates           100% |=========================| 1.1 kB
00:00
updates                   100% |=========================| 1.1 kB
00:00
testing-updates           100% |=========================|  951 B
00:00
Reading repository metadata in from local files
Available Packages
curl.i386                               7.15.0-1
rawhide-updates
```

Získávání popisu balíčku se provádí příkazem info a jménem balíčku o kterém si přejeme vědět více. Toto dostaneme pokud na předchozí balíček použijeme příkaz info:

```
  $ yum info curl

Setting up repositories
rawhide-updates           100% |=========================| 1.1 kB
00:00
updates                   100% |=========================| 1.1 kB
00:00
testing-updates           100% |=========================|  951 B
00:00
Reading repository metadata in from local files
```

```
Available Packages
Name   : curl
Arch   : i386
Version: 7.15.0
Release: 1
Size   : 253 k
Repo   : rawhide-updates
Summary: A utility for getting files from remote servers (FTP, HTTP,
and others).
Description:
cURL is a tool for getting files from FTP, HTTP, Gopher, Telnet, and
Dict servers, using any of the supported protocols. cURL is designed
to work without user interaction or any kind of interactivity. cURL
offers many useful capabilities, like proxy support, user
authentication, FTP upload, HTTP post, and file transfer resume.
```

Výstup z tohoto příkazu nám ukazuje detaily balíčku, jeho velikost a popis využití.

Jednou z hlavních předností Yum jsou jeho vyhledávací možnosti. Yum dovoluje vyhledávat Balíčky podle klíčového slova, jména balíčku i cesty. Navíc při vyhledávání balíčků lze použít možnost „whatprovides“ k vyhledání balíčku obsahující specifický spustitelný soubor.

```
$ yum whatprovides /etc/yum.conf

Searching Packages:
Setting up repositories
updates-released          100% |=========================|  951 B
00:00
extras                    100% |=========================| 1.1 kB
00:00
base                      100% |=========================| 1.1 kB
00:00
Reading repository metadata in from local files
primary.xml.gz            100% |=========================| 977 kB
00:01
extras    : ##################################################
2714/2714
Added 8 new packages, deleted 0 old in 6.65 seconds

yum.noarch                                  2.3.2-7               base
Matched from:
/etc/yum.conf

yum.noarch                                  2.4.0-0.fc4
updates-released
Matched from:
/etc/yum.conf

yum.noarch                                  2.4.0-0.fc4
installed
Matched from:
/etc/yum.conf
```

Zde vidíme že /ect/yum.conf je součástí balíčku yum.noarch.

Mnoho balíčků které se nachází v linuxových distribucích nemůže fungovat samostatně, obsahují proto jednu nebo více závyslostí, tedy balíčků bez kterých nemohou

fungovat. Prefektním příkladem je Apache web server. Apache potřebuje ke své funkci mnoho dalších jako například PHP balíček nebo balíček na podporu SSL. Správa těchto závyslostí je obor kde yum opravdu září, nejenom že dokáže automaticky detekovat závyslosti a vemi efektivně je řešit, dovoluje uživateli i prohlédnout si list závyslostí pomocí příkazu deplist:

```
$ yum deplist curl

Finding dependencies:
Setting up repositories
updates-released          100% |=========================|  951 B
00:00
extras                    100% |=========================| 1.1 kB
00:00
base                      100% |=========================| 1.1 kB
00:00
Reading repository metadata in from local files
package: curl.i386 7.13.1-3
  dependency: libdl.so.2
   provider: glibc.i386 2.3.5-10
   provider: glibc.i686 2.3.5-10
   provider: glibc.i386 2.3.5-10.3
   provider: glibc.i686 2.3.5-10.3
  dependency: libkrb5.so.3
   provider: krb5-libs.i386 1.4-3
   provider: krb5-libs.i386 1.4.1-5
  dependency: libc.so.6(GLIBC_2.1)
   provider: glibc.i386 2.3.5-10
   provider: glibc.i686 2.3.5-10
   provider: glibc.i386 2.3.5-10.3
   provider: glibc.i686 2.3.5-10.3
  dependency: libcrypto.so.5
   provider: openssl.i686 0.9.7f-7
   provider: openssl.i386 0.9.7f-7
   provider: openssl.i386 0.9.7f-7.10
   provider: openssl.i686 0.9.7f-7.10
  dependency: libc.so.6
   provider: glibc.i386 2.3.5-10
   provider: glibc.i686 2.3.5-10
   provider: glibc.i386 2.3.5-10.3
   provider: glibc.i686 2.3.5-10.3
```

a ještě mnoho dalšího, závyslosti jsou opravdu velmi rozsáhlá část práce balíčkovacích systémů a jejich správná funkce zajišťuje pohodlí uživatele protože se nemusí sám starat o manuální získávání všech náležitostí k běhu jím požadovaného programu bez probémů.

Instalace balíčků je základní funkcí každého manažeru. K tomu také patří i odinstalace a řešení závyslostí které mohou být různé v závyslosti na verzi balíčku. Yum instaluje prostým příkazem install a jeho argumenty může být jeden nebo více balíčků. Během instalace se musí vyřešit rekurzivně všechny závyslosti a závyslosti závyslostí. Po jejich vyřešení se sestaví transakce. Takto vypadá proces instalace balíčku curl:

```
$ yum install curl

Setting up Install Process
```

```
Setting up repositories
updates-released          100% |=========================|  951 B
00:00
extras                    100% |=========================| 1.1 kB
00:00
base                      100% |=========================| 1.1 kB
00:00
Reading repository metadata in from local files
Parsing package install arguments
Resolving Dependencies
--> Populating transaction set with selected packages. Please wait.
---> Downloading header for curl to pack into transaction set.
curl-7.13.1-3.i386.rpm    100% |=========================|  10 kB
00:00
---> Package curl.i386 0:7.13.1-3 set to be updated
--> Running transaction check

Dependencies Resolved

================================================================================
========
 Package                 Arch       Version          Repository
Size
================================================================================
========
Installing:
 curl                    i386       7.13.1-3         base
262 k

Transaction Summary
================================================================================
========
Install      1 Package(s)
Update       0 Package(s)
Remove       0 Package(s)
Total download size: 262 k
Is this ok [y/N]: y
Downloading Packages:
(1/1): curl-7.13.1-3.i386 100% |=========================| 262 kB
00:00
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing: curl                         ##########################
[1/1]

Installed: curl.i386 0:7.13.1-3
Complete!
```

Odinstalace příkazem uninstall se řeší velmi podobně, zpětným procesem instalace včetně řešení závyslostí, tentokrát zda se neodstraňují balíčky nutné k funkci zbylých programů v systému.

```
$ yum remove curl

Setting up Remove Process
Resolving Dependencies
--> Populating transaction set with selected packages. Please wait.
```

```
---> Package curl.i386 0:7.13.1-3 set to be erased
--> Running transaction check

Dependencies Resolved

================================================================================
========
 Package                 Arch       Version          Repository
Size
================================================================================
========
Removing:
 curl                    i386       7.13.1-3         installed
521 k

Transaction Summary
================================================================================
========
Install      0 Package(s)
Update       0 Package(s)
Remove       1 Package(s)
Total download size: 0
Is this ok [y/N]: y
Downloading Packages:
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Removing  : curl                          ##########################
[1/1]

Removed: curl.i386 0:7.13.1-3
Complete!
```

Jednou z nejlepších funcí Yum je jeho schopnost porovnat instalované balíčky s jejich verzí ve vzdáleném repozitáři. Porovnání je provedeno pro všechny nainstalované balíčky příkazem check-update. Pokud budeme chtít lze všechny neaktualizované balíčky aktualizovat příkazem update. Update postupuje naprosto stejně jako standartní instalace vyřešením závyslostí a sestavením transakcí. Pokud chceme aktualizovat pouze vybrané balíčky musíme je uvést jako argument pro funkci update. Vzhledem k délce výstupu je příklad zkrácen.

```
$ yum check-update

Setting up repositories
updates-released          100% |=========================|  951 B
00:00
extras                    100% |=========================| 1.1 kB
00:00
base                      100% |=========================| 1.1 kB
00:00
Reading repository metadata in from local files
primary.xml.gz            100% |=========================| 337 kB
00:01
updates-re: ##################################################
989/989
Added 105 new packages, deleted 83 old in 4.52 seconds
```

```
primary.xml.gz            100% |=========================| 977 kB
00:00
extras    : ##################################################
2714/2714
Added 8 new packages, deleted 0 old in 6.60 seconds

bind.i386                                 24:9.3.1-14_FC4
updates-released
bind-libs.i386                            24:9.3.1-14_FC4
updates-released
bind-utils.i386                           24:9.3.1-14_FC4
updates-released
esound.i386                               1:0.2.36-0.fc4.1
updates-released
esound-devel.i386                         1:0.2.36-0.fc4.1
updates-released
gawk.i386                                 3.1.4-5.3
updates-released
gdb.i386                                  6.3.0.0-1.84
updates-released
logwatch.noarch                           7.0-1.fc4
updates-released
mutt.i386                                 5:1.4.2.1-4.FC4
updates-released
pam.i386                                  0.79-9.6
updates-released
pam-devel.i386                            0.79-9.6
updates-released
sudo.i386                                 1.6.8p8-2.3
updates-released


$ yum update

Setting up Update Process
Setting up repositories
updates-released          100% |=========================|  951 B
00:00
extras                    100% |=========================| 1.1 kB
00:00
base                      100% |=========================| 1.1 kB
00:00
Reading repository metadata in from local files
Resolving Dependencies
--> Populating transaction set with selected packages. Please wait.
---> Downloading header for bind to pack into transaction set.
bind-9.3.1-14_FC4.i386.rp 100% |=========================|  39 kB
00:00
---> Package bind.i386 24:9.3.1-14_FC4 set to be updated
---> Downloading header for gawk to pack into transaction set.
gawk-3.1.4-5.3.i386.rpm   100% |=========================|  17 kB
00:00

...

---> Package gdb.i386 0:6.3.0.0-1.84 set to be updated
--> Running transaction check

Dependencies Resolved
```

```
================================================================================
========
 Package                 Arch       Version          Repository
Size
================================================================================
========
Updating:
 bind                    i386       24:9.3.1-14_FC4  updates-released
532 k
 bind-libs               i386       24:9.3.1-14_FC4  updates-released
779 k
 bind-utils              i386       24:9.3.1-14_FC4  updates-released
146 k
 esound                  i386       1:0.2.36-0.fc4.1  updates-
released  127 k
 esound-devel            i386       1:0.2.36-0.fc4.1  updates-
released   31 k
 gawk                    i386       3.1.4-5.3        updates-released
1.7 M
 gdb                     i386       6.3.0.0-1.84     updates-released
2.7 M
 logwatch                noarch     7.0-1.fc4        updates-released
217 k
 mutt                    i386       5:1.4.2.1-4.FC4  updates-released
1.1 M
 pam                     i386       0.79-9.6         updates-released
1.9 M
 pam-devel               i386       0.79-9.6         updates-released
87 k
 sudo                    i386       1.6.8p8-2.3      updates-released
185 k

Transaction Summary
================================================================================
========
Install      0 Package(s)
Update      12 Package(s)
Remove       0 Package(s)
Total download size: 9.5 M
Is this ok [y/N]: y
Downloading Packages:
(1/12): bind-9.3.1-14_FC4 100% |=========================| 532 kB
00:02
(2/12): gawk-3.1.4-5.3.i3 100% |=========================| 1.7 MB
00:03

...

(12/12): gdb-6.3.0.0-1.84 100% |=========================| 2.7 MB
00:05
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Updating  : bind-libs                    ######################### [
1/24]
  Updating  : pam                          ######################### [
2/24]

...
```

```
  Cleanup   : bind                          ######################
[13/24]
  Cleanup   : gawk                          ######################
[14/24]

...

  Cleanup   : gdb                           ######################
[24/24]

Updated: bind.i386 24:9.3.1-14_FC4 bind-libs.i386 24:9.3.1-14_FC4
bind-utils.i386 24:9.3.1-14_FC4 esound.i386
1:0.2.36-0.fc4.1 esound-devel.i386 1:0.2.36-0.fc4.1 gawk.i386
0:3.1.4-5.3 gdb.i386 0:6.3.0.0-1.84 logwatch.noarch
0:7.0-1.fc4 mutt.i386 5:1.4.2.1-4.FC4 pam.i386 0:0.79-9.6 pam-
devel.i386 0:0.79-9.6 sudo.i386 0:1.6.8p8-2.3
Complete!
```

Yum si pro rychlejší práci udržuje cache hlaviček v souboru /var/cache/yum. Tento soubor používáním nabyde na velikosti a proto yum nabízí možnost jeho bezpečného vyčistění příkazem clean.

```
$ yum clean all

Cleaning up Everything
251 headers removed
251 packages removed
6 metadata files removed
0 cache files removed
3 cache files removed
```

Opět pokud je nutné vyčistit pouze některé hlavičky opět musí být uvedeny jako argumenty, příkaz bez argumentů vyčistí celé cache. Je doporučováno provádět čistění pravidelně kvůli šetření místa na disku.

Yum nabízí ještě mnoho dalsích funkcí které lze použít pro snadnější zprávu nebo užívání, pro účely mé bakalářské práce však stačí tyto, protože jsou to prvky obsluhující instalaci balíčků a vzajemnou integritu systému.

2.Pluginy

Od verze 2.0 youm umožňuje programování rozšíření v Pythonu které dovolují měnit chování yumu. Balíček yum-utils obsahuje příkazy které používají yum API a mnoho dalsích pluginů.

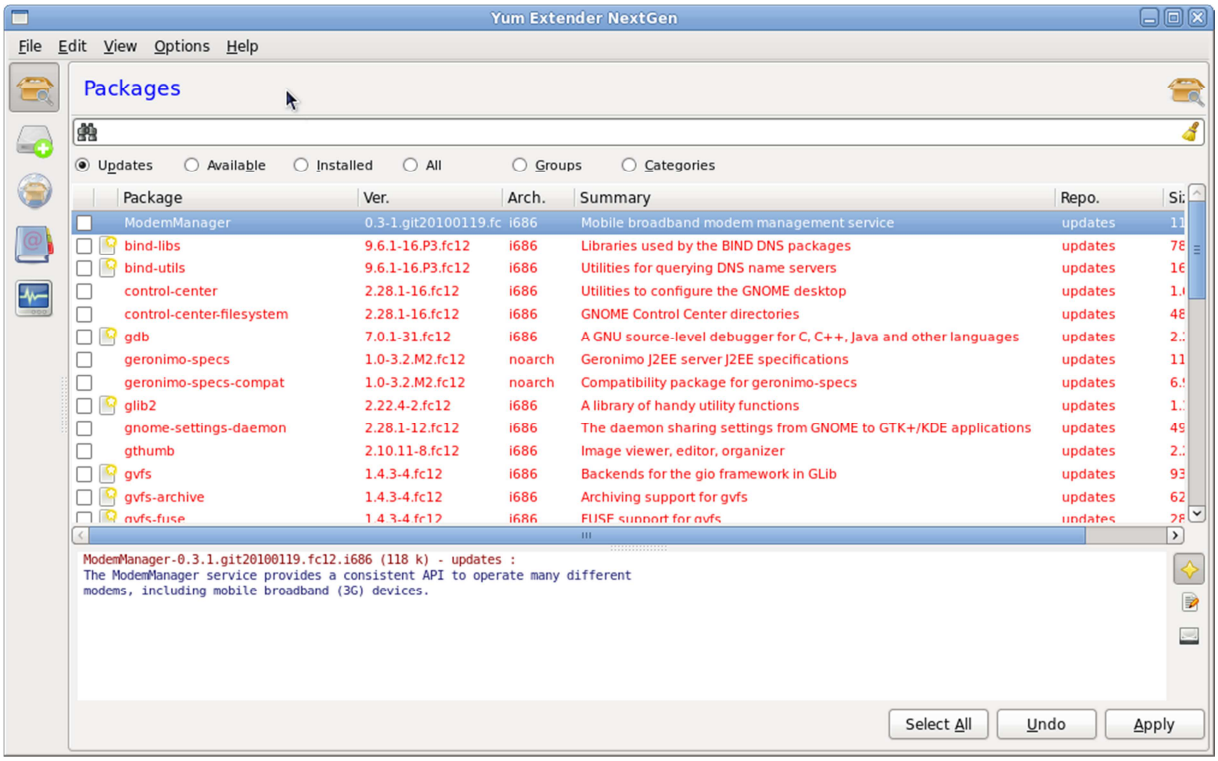
2.1 DNF

DNF je další vývojovou větví YUM pro fedoru 18, Používá knihovnu Hawkey jako back-end. Hlavním cíle je vyšší výkon a lepší práce s pamětí. Také stanovuje pevné API definice pro pluginy a následné rozšíření.

3 Existující uživatelské rozhraní YUM

3.1 YUM extender

Yum extender je velmy příjemné a intuitivní uživatelské rozhraní pro systémy Fedora, RHEL, CentOS a další RPM linux systémy. Jedná se o jednoduchý nástroj napsaný taktéž v Pythonu a používající PyGTK toolkit pro GUI komponenty.



Obrázek 1: YUM Extender

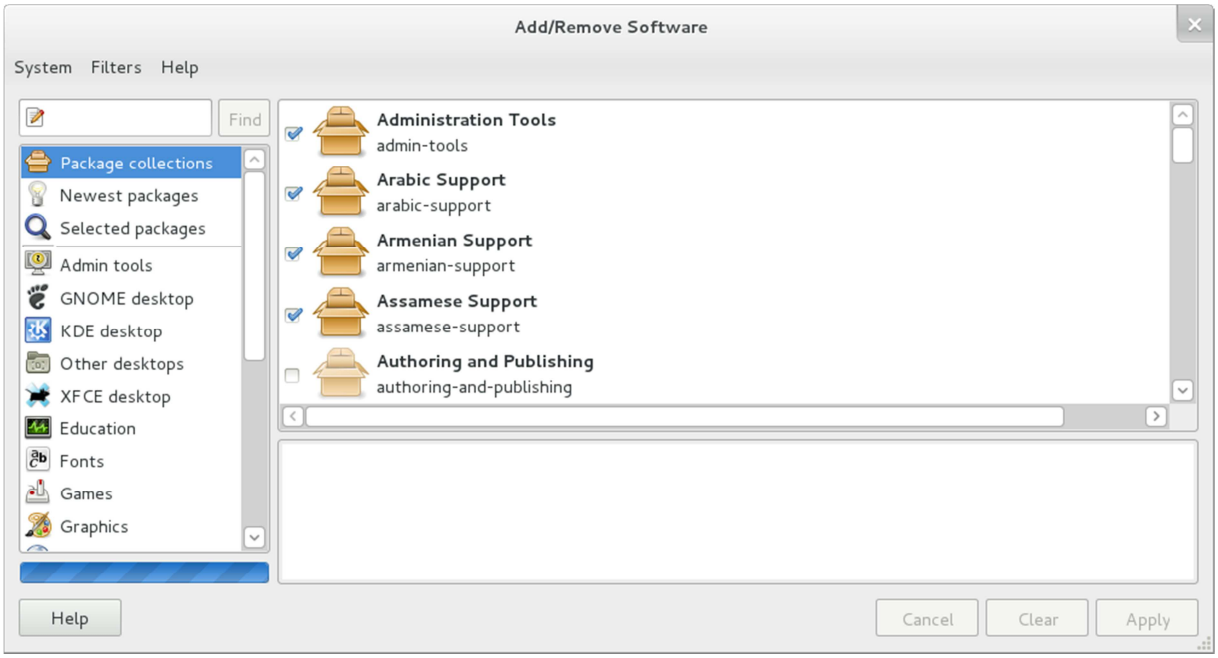
Hlavní cíle Yumexu jsou

- Vytvořit pokročilé rozhraní pro zkušené uživatele i nováčky
- Jednoduchá aktualizace, instalace i odinstalace aplikace
- Jednoduché vyhledávání balíčků
- Dát možnost nahlédnout co se děje v pozadí programu
- Ukázat sílu YUM
- Poskytnout přístup k pokročilým funkcím Yumu lehkou cestou

Z yum extenderu jsem čerpal inspiraci hlavně v podobě funkcí a možností mého rozhraní.

3.2 PackageKit

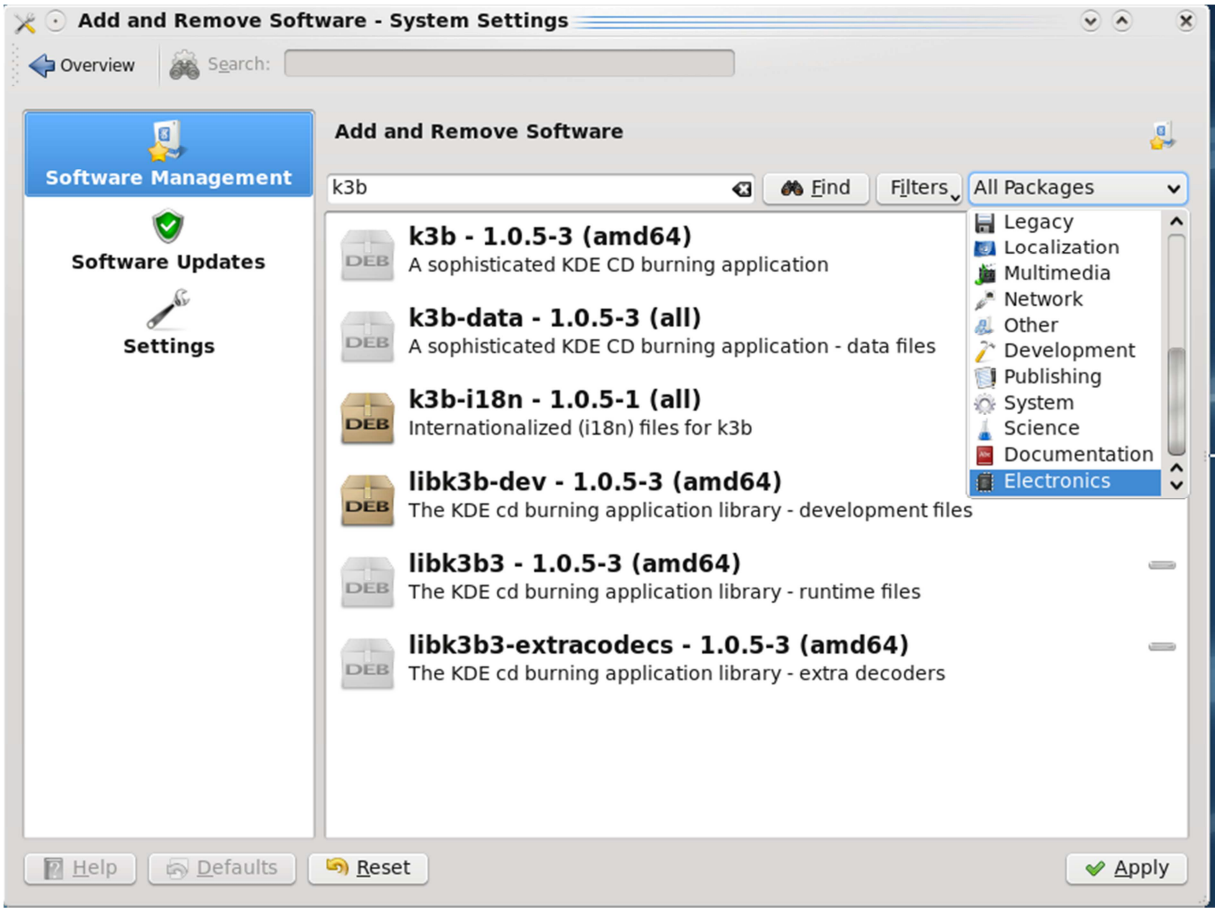
Původně byl vyvinut Richardem Hugesem a jeho teamem a poprvé byl využit jako základní nadstavbu YUM v operačním systému Fedora 9. Od té doby prošel mnoha vylepšnímy v systémech Fedora 10 a fedora 11. Jedná se o soubor softwarových aplikací vyvinutých pro několik balíčkovacích systémů. Je mnohoplatformní, ikdyž je primárně určen na Linuxové distribuce. Využívá softwaroví knihovny D-Bus projektu a PolicyKit na zajištění meziprocesové komunikace a privilegií. PackageKit funguje jako systémvý démon což mu umožňuje abstrakci a tedy práci na rozdílných systémech. Sám packageKit má 3 nadstavby: gnome-packagekit pro rozhraní GNOME, Appear pro rozhraní KDE a pkcon pro příkazovou řádku.



Obrázek 2:GnomePackageKit

Příklady funkcí package kitu

- Authorizace a práva pomocí PoliciKit
- Víceuživatelský systém – nedovolí vypnutí pokud probíhá kritická část transakce
- Nenahrazuje existující systémový manažer balíčků
- Instalace z lokálních úložišť i vzdálených zdrojů



Obrázek 3: KPackageKit

4 Použité nástroje

4.1 Ncurses

Ncurses, jinak také new curses jsou programovací knihovna která přináší API co dovoluje programátorovi psát textové uživatelské rozhraní nezávyslé na terminálu. Vytváří se tak „polo-grafické“ rozhraní které beží na emulátoru terminálu. Také dokáže přizpůsobit obraz aby minimalizoval latenci při použití vzdálené správy. První knihovny curses byli vyvinuty na univerzitě v Berkeley v Californii pro operační sytémy BSD kolem roku 1980. Původně využívaly termcap knihovny co byly použité v jiných programech, například v vi editoru. Úspěch BSD curses knihoven donutil Bellovi Laboratoře k vývoji vylepšených curses v System III a System V Unixech. Tyto knihovny byly mnohem výkonnější a místo termcap využívaly terminfo. Kolem roku 1982, Pavel Curtis začal pracovat na freeware klonu pojmenovaném pcurses. Pcurses byly dále zlepšeny když vývoj převzal zeyd Ben-Halim v roce 1991. Nové Ncurses byly vydány v Listopadu 1993 ve verzi 1.8.1. V současnosti se od roku 1996 stará o vývoj Thomas E. Dickey. Mnoho příkazů ncurses lze lehce přepsat pro starší curses pro případnou nutnost používat starší systém Unix. Pro použití ncurses jsem se rozhodl protože nejlépe splňuje mé potřeby k vypracování zadání. Stavba rozhraní je nenáročná na pochopení a lze dosáhnout velmi uspokojivých výsledků s relativně málo příkazy.

4.2 Python

Oběktově orientovaný skriptovací jazyk. Python je navržen aby bylo možné v něm programovat rozsáhlé plnohodnotné aplikace včetně grafického rozhraní. Při psaní programu může programátor používat několik paradigmat: Objektově orientované, procedurální, ale i funkcionální. Díky této možnosti lze python použít téměř pro všechny účely vhodným výběrem paradigma programování. Jednoduchos jazyka Python z něj dělá i velmi vhodný programovací jazyk pro začátečníky neboť byl při tvorbě inspirován Jazykem ABC, který byl pro výuku programování přímo vytvořen.

Významnou vlastností Pythonu je produktivnost z hlediska rychlosti psaní programů. Týká se to jak nejjednodušších programů, tak aplikací velmi rozsáhlých. U jednoduchých programů se tato vlastnost projevuje především stručností zápisu. U velkých aplikací je produktivnost podpořena rysy, které se používají při programování ve velkém, jako jsou například přirozená podpora jmenných prostorů, používání výjimek, standardně dodávané prostředky pro psaní testů (unit testing) a dalšími. S vysokou produktivností souvisí dostupnost a snadná použitelnost široké škály knihovních modulů, umožňujících snadné řešení úloh z řady oblastí.

Python se snadno vkládá do jiných aplikací, kde pak slouží jako jejich skriptovací jazyk. Tím lze aplikacím psaným v kompilovaných programovacích jazycích dodávat chybějící pružnost. Jiné aplikace nebo aplikační knihovny mohou naopak implementovat rozhraní, které umožní jejich použití v roli pythonovského modulu. Jinými slovy, pythonovský program je může využívat jako modul dostupný přímo z jazyka Python takzvaný extending.Programování v Pythonu klade velký důraz na produktivitu práce programátora.

Python 3 je vyvíjen s důrazem na pragmatičnost. To znamená, že vývoj jeho verzí je spíše evoluční. Přirozeným důsledkem takového přístupu je i zpětné hodnocení dobrých a horších vlastností jazyka. Jeho výsledkem byl projekt Python 3000 (Py3k) jako základ vývoje přelomové verze Python 3.

Stabilní verze Python 3.0 byla vypuštěna v 3. prosince 2008. Je zpětně nekompatibilní. Přechodovou verzí mezi Python 2.x a Python 3.0 představuje Python 2.6 (varování při použití syntaxe, která nebude ve verzi 3.0 platná). Současně byly vyvinuty nástroje pro usnadnění konverze starších zdrojových textů do podoby pro verzi Python 3.0.

Velmi významnou změnou je důsledné oddělení abstrakcí řetězec a posloupnost bajtů. Řetězce se důsledně mění na typ unicode. Pro posloupnosti bajtů je zaveden nový typ bytes (immutable) a bytearray (mutable). Při vzájemném převodu mezi řetězcem a posloupností bajtů je nutné vždy uvádět požadované kódování. To by mělo vést k důslednému vyřešení problémů, které se projevovaly v souvislosti se znaky národních abeced. Python 3 je již obsažen v základní distribuci většiny běžných systémů Linux

Hlavní principy které při návrhu jazyka prosazuje jeho tvůrce Guido van Rossum stručně shrnul dlouholetý pythnový programátor Tim Peters jako Zen of Python – 20 aforismů o Pythonu z nichž bylo zapsáno pouze 19.

he Zen of Python, by Tim Peters	**Volný překlad**
Beautiful is better than ugly.	Krásný je lepší než ošklivý.
Explicit is better than implicit.	Explicitní je lepší než implicitní.
Simple is better than complex.	Jednoduchý je lepší než složitý.
Complex is better than complicated.	Složitý je lepší než komplikovaný.
Flat is better than nested.	Plochý je lepší než zanořený.
Sparse is better than dense.	Řídký je lepší než hustý.
Readability counts.	Na čitelnosti záleží.
Special cases aren't special enough to break the rules.	Zvláštní případy nejsou dost zvláštní na to, aby ospravedlnily porušení pravidel.
Although practicality beats purity.	Ačkoliv praktičnost vyhrává nad čistotou.
Errors should never pass silently.	Chyby by nikdy neměly projít potichu.
Unless explicitly silenced.	Pokud nejsou záměrně zamlčeny.
In the face of ambiguity, refuse the temptation to guess.	Pokud se setkáš s nejednoznačností, odolej pokušení odhadovat.
There should be one—and preferably only one—obvious way to do it.	Měl by existovat jeden — a pokud možno pouze jeden — zřejmý způsob jak to udělat.
Although that way may not be obvious at first unless you're Dutch.	Ačkoliv tento způsob nemusí být hned zřejmý, pokud nejsi Holanďan.
Now is better than never.	Teď je lepší než nikdy.
Although never is often better than **right** now.	Ačkoliv nikdy je často lepší než **právě** teď.
If the implementation is hard to explain, it's a bad idea.	Pokud lze implementaci vysvětlit jen s obtížemi, jde o špatnou myšlenku.
If the implementation is easy to explain, it may be a good idea.	Pokud lze implementaci vysvětlit snadno, mohla by to být dobrá myšlenka.
Namespaces are one honking great idea — let's do more of those!	Jmenné prostory jsou jednou z velkých myšlenek — jen víc takových!

Anlický originál textu se vypíše pokud uživatel zadá příkaz >>> import this.

5 Vlastní Implementace

5.1 API

API neboli Application programming interface je rozhraní pro programování aplikací. Jde o sbírku tříd, funkcí a procedur kterí můžeme využívat k dosažení funkčnosti našeho programu. V našem případě se jedná o funkce YUM ke kterým hodlám programovat uživatelské rozhraní aniž bych musel složitě tyto funkce a procedury znovu vymýšlet.
V programu budu pouze volat funkce API a o správnost vykonání příkazu se už postará samotný YUM. Při instalaci balíčku je nutné dohledat jeho závyslosti na jiných balíčcích pro zajištění jeho plné funkčnosti a rekurzivně i funkčnosti všech dalších stažených balíčků, toto všechno vyřeší YUM sám když moje rozhraní bude volat funkci instalovat balíček.

5.2 Příkazy

- Nainstaluj – Nainstaluje právě vybraný balíček ve sloupci balíčků k dispozici nebo přeinstaluje právě vybraný balíček ve sloupci Nainstalováno
- Odinstaluj – Odinstaluje vybraný balíček ve sloupci Nainstalováno, ve sloupci K Dispozici není tento příkaz aktivní

5.3Možnosti

Program umožňuje plnou funkčnost jako rozhraní pro YUM, je naprogramován způsobem který dovoluje snadnou rozšiřitelnost doprogramováním funkcí případně i pluginů. Vleké množství funkcí je pouze naznačeno neboť jejich implementaci jsem nestihl v termínu, jedná se zejména o funkce na úplné překreslení po instalaci a odinstalaci a vznik fragmentů vinou neaktualizování daného znaku.

5.4 Využití

Je intuitivní a snadno jej pochopí i naprostý laik v oboru. Díky velmi malé závislosti na ostatním programovém vybavení systému, python má každý redHat linux ve své základní instalaci, jej lze snadno použít k opravě nainstalovaných balíčků například při zhroucení grafického rozhraní. Výhodou textového rozhraní je i možnost instalovat systém naprosto bez grafického rozhraní a tím ušetřit zdroje například na clusterech. Textové rozhraní funguje spolehlivě za všech okolností pokud má uživatel přístup do konzole systému. K programu jsem přistupoval s cílem vytvořit pokud možno nejjistěji fungující interface pro dobu nouze uživatele, proto je rozhraní jednoduché a nemá mnoho ovládacích prvků. Mezi tyto patří obnovení seznamů balíčků, instalace odinstalace a zobrazení informací o balíčku.
V budoucích verzích bych se rád zaměřil na vyřešení překreslování rozhraní, aby nevznikaly fragmenty v místech kde se nemění znaky

5.5Spuštění

Takto vypadá program po spuštění a načtení seznamů, v levém sloupci obsahuje seznam balíčků aktuálně k dispozici pro možnou instalaci. Instalace se provádí příkazem pro API kde YUM vykoná vše potřebné pro instalaci balíčku jako jeho uložení, vyhledání závislostí a další úkony potřebné pro bezproblémovou instalaci. Po nainstalování se provede obnovení obou seznamů. V levém sloupci je seznam aktuálně nainstalovaných balíčků. Odinstalace se provádí vemi podobně jako instalace příkazem pro API. Program nepodporuje myš a proto jsou vechny příkazy pro přehlednost uvedeny v dolní části jako reference. Pokud v dalších verzích bude použití myši implementováno budou sloužit jako tlačítka.

Program je nutné spouštět jako Administrátor příkazem „sudo“ aby měl program pravomoce pro instalaci a odinstalaci balíčků..

Během spuštění program načítá oba seznamy a po každé transakci tyto seznamy obnovuje. Seznamy lze obnovit i manuálním příkazem uživatele pomocí klávesy F4

```
curses.init_pair(1, curses.COLOR_RED, curses.COLOR_BLUE)

self.DIM_RED = curses.color_pair(1)

self.RED = curses.color_pair(1) | curses.A_BOLD

self.REVERSE = curses.A_REVERSE | curses.A_DIM | curses.A_BOLD
```

Ukázka kódu 1: Inicializace barev

Ukázka kódu k Inicializaci použitých barev v programu, použil jsem snadno rozpoznatelné barevné schéma používané v ranných dobách textových rozhraní pro MS-DOS jako manažer m602 případně Midnight Commanderu pro linux. Základní barva pozadí je modrá a text žlutý. Ncurses pracují s páry barev aby bylo možné text označit, tedy změnit barvu znaků a nechat pozadí stejné jako okolí, nebo jej zvýraznit, změnit barvu i pozadí, nejlépe velmi kontrastně. Vybrané položky v programu se barvý červeně . Právě ovládaný prvek je zvýrazněn jinou barvou písma.

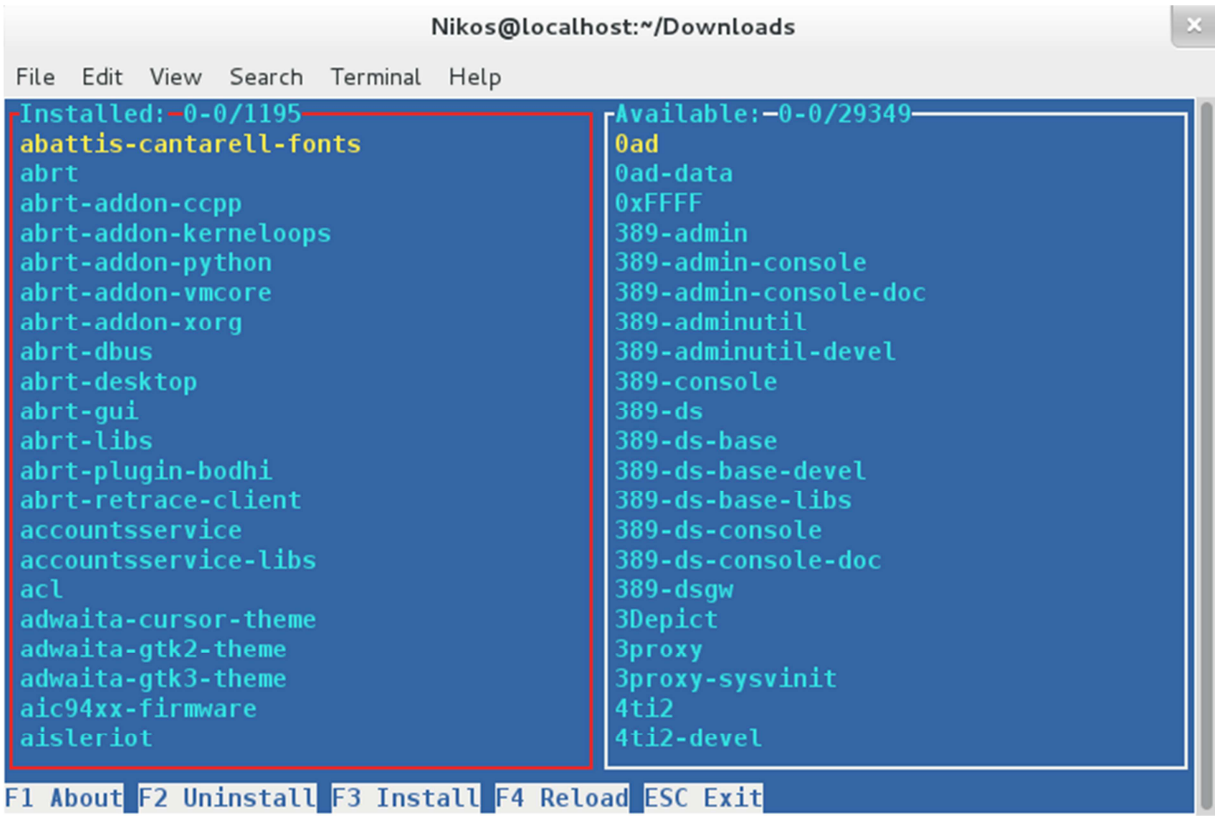
Funkce obstarávající vlastní zobrazení rozhraní a jeho aktualizaci. Skládá se z tří funkcí na vykreslení oken a funkce na aktualizaci, refresh, po každé uživatelské akci se překresluje právě použitý znak, vzhledem k náročnosti obnovování celého textového rozhraní se jedná o významnou úsporu prostředků..

Funkce pro detekci použitých kláves funkcí getch() a jejich překlad pro knihovnu curses. Vzhledem k malému počtu funkčních kláves jsem použil rozcestník z podmínek kde je každý nerozpoznaný znak zahozen, pokud by byl počet funkčních kláves vyšší, například celá klávesnice, rozhodně bych postupoval jinou metodou.

Funkce pro vykreslení oken programu, každé okno je konstruováno jako samostatná instance třídy self. Tyto funkce vykonstruují každé okno podle zadaných parametrů jako tvar, rozměry ve znacích, barvy a umístění vzhledem k počátku. Zde je kód pro vytvoření okna installed a jeho zabarvení pokud je označené jako aktivní, tedy nalézá se v něm kurzor.

```
def installed_window(self):
        '''...'''
     self.Installed = curses.newwin(self.window_max_y - 1, 40,
     0, 0)
     if (self.active_window == self.INSTALLED):
           self.Installed.bkgd(self.RED)
     else:
           self.Installed.bkgd(self.WHITE)
```

Ukázka kódu 2: okno installed



Obrázek 4:Po spuštění

Po vytvoření oken se tato okna naplní seznamem balíčků, podle druhu okna připravených k instalaci nebo nainstalovaných a tedy připravených k odinstalaci. Po vytvoření hlavního okna dojde k načtení seznamu položek a jejich zobrazení.

```
def show_data_installed(self):
    '''Display loaded data to the installed subwindow'''
    data_width = 38
    (sub_win_y, sub_win_x) = self.Installed.getmaxyx()
    sub_win_y -= 2
    sub_win_x -= 2

    # size of inner 'panel'
    pad_y = max(sub_win_y, len(self.installed_data)+1)
    pad_x = max(sub_win_x, 300)
    top = self.installed_top
    bottom = self.installed_top + self.N_LINES
    for (index, line) in enumerate(self.installed_data[top:bottom]):
      line_num = self.installed_top + index
      if (line_num in self.installed_marked):
        if (index != self.active_installed):
          self.Installed.addstr(index + 1, 1, line[0][:data_width],
self.RED)
        else:
          self.Installed.addstr(index + 1, 1, line[0][:data_width],
self.WHITE)
      else: # not marked
        if (index != self.active_installed):
          self.Installed.addstr(index + 1, 1, line[0][:data_width],
self.CYAN)
        else:
          self.Installed.addstr(index + 1, 1, line[0][:data_width],
self.YELLOW)
    return
```

Ukázka kódu 3: načtení seznamu installed

Zde je kód pro načtení seznamu nainstalovaných balíčků. Pokud se v okně nalézá kurzor, tedy aktivní řádek, dojde k zabarvení okraje okna (viz obr. 5).

5.6 Instalace Balíčků

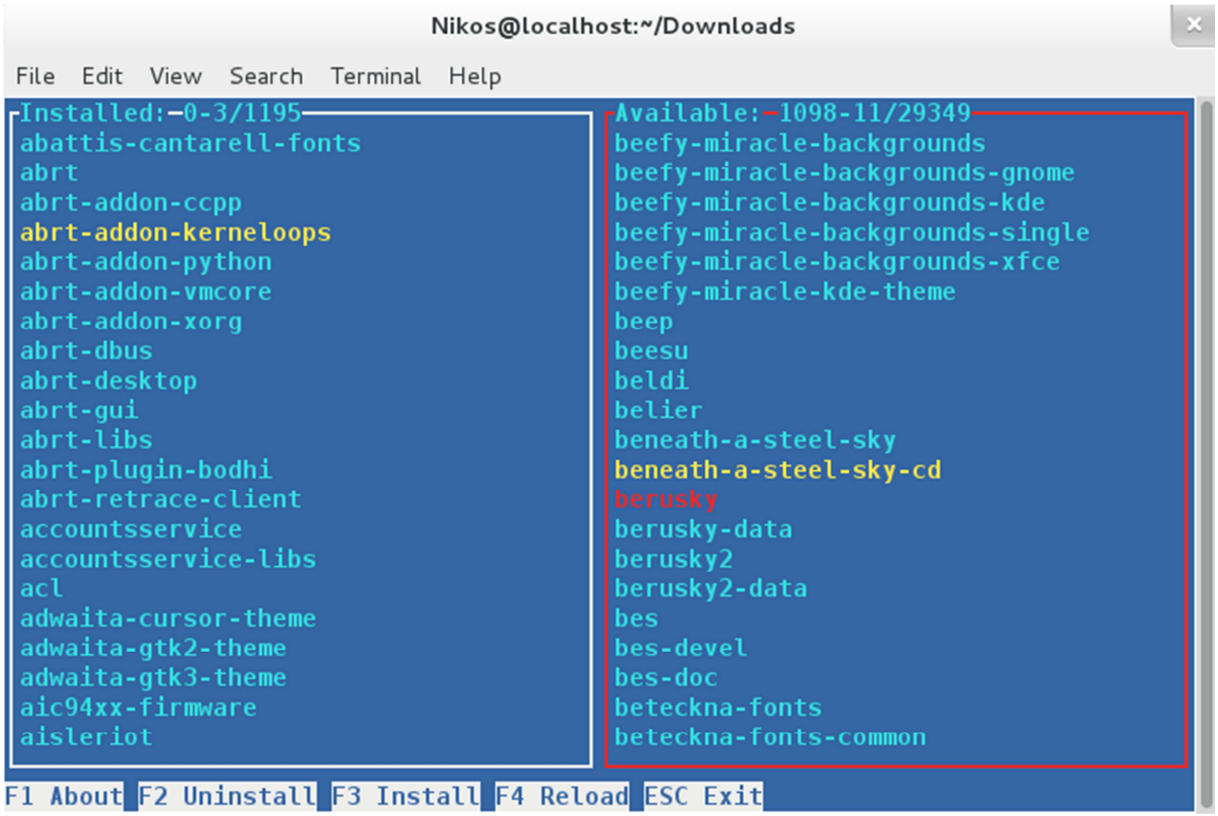
Instalaci balíčků jako takovou obstarává YUM voláním funkcí jeho API. Yum také řeší závyslosti voláním funkce *yum_base.resolveDeps()*. Celý proces tedy vypadá takto:

```
 def install_marked(self):
    '''Method for installing previously marked (available)
packages
      ...Under Construction...
    '''
    install_list = list()
    for index in sorted(self.available_marked):
      install_list.append(self.available_data[index][1])
    for po in install_list:
      self.yum_base.install(po)

    self.pop_up()

    self.yum_base.resolveDeps()
    self.yum_base.buildTransaction()
    self.yum_base.processTransaction()
    self.Installed.noutrefresh()
    self.Available.noutrefresh()
    self.Command.noutrefresh()
    self.stdscr.refresh()
```

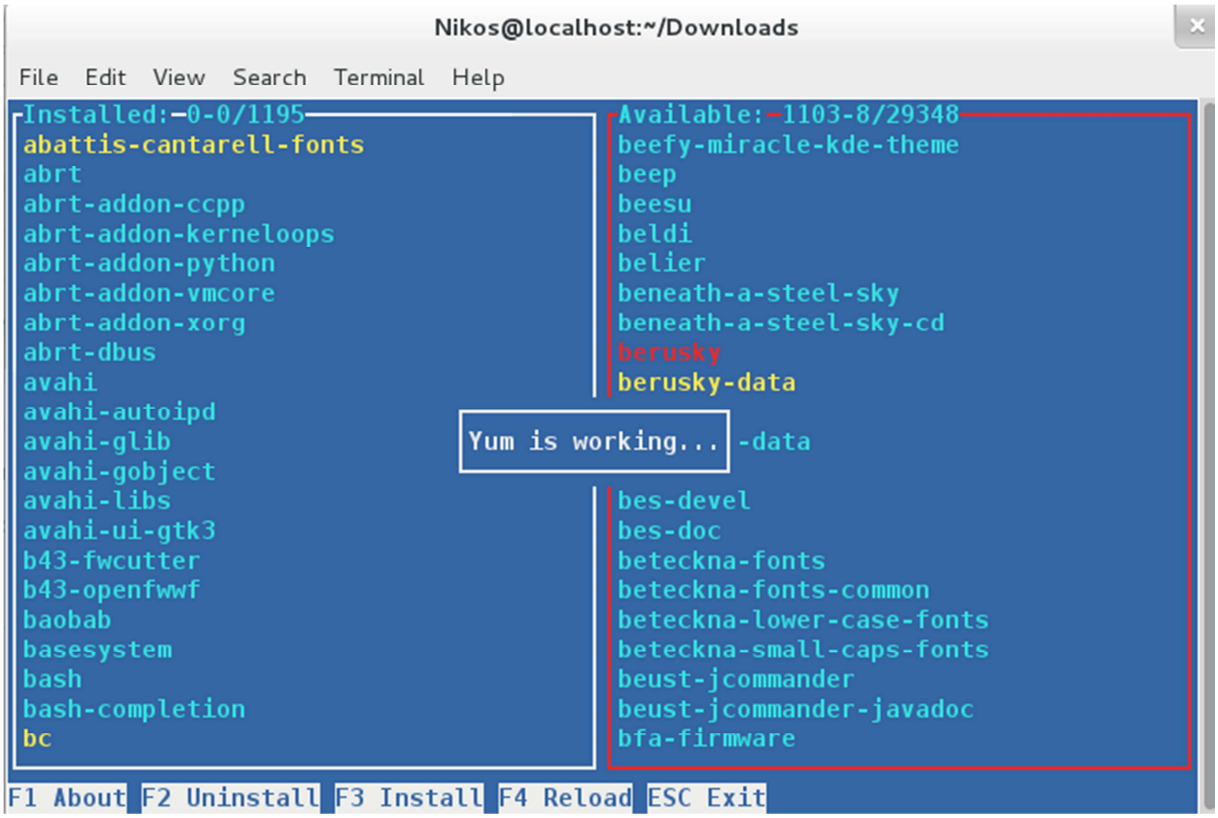
Ukázka kódu 3: instalace balíčku

Po načtení seznamu označených položek v okně předám pomocí funkce *yum_base.install(po)* kde jako argument předávám názvy balíčků k instalaci. Funkce *self.pop_up()* je volání dialogového okna pro varování uživatele, že YUM pracuje a interface nebude reagovat. Funkce potom vyřeší závyslosti pomocí *yum_base.resolveDeps()*. *yum_base.buildTransaction()* a *yum_base.processTransaction()* sestaví a provedou transakci, tedy instalaci jednoho či více balíčků a jejich závyslostí. Poté dojde k obnovení celého rozhraní s nově aktualizovanými seznamy, tento krok je významný protože jinak by mohlo dojít k nekonzistenci seznamů a tedy posílání chybných požadavků.

Odinstalace probíhá naprosto stejným způsobem, pouze je volána funkce *yum_base.remove(po)* namísto *yum_base.install(po)* Argumentem je opět seznam balíčků k odinstalaci.



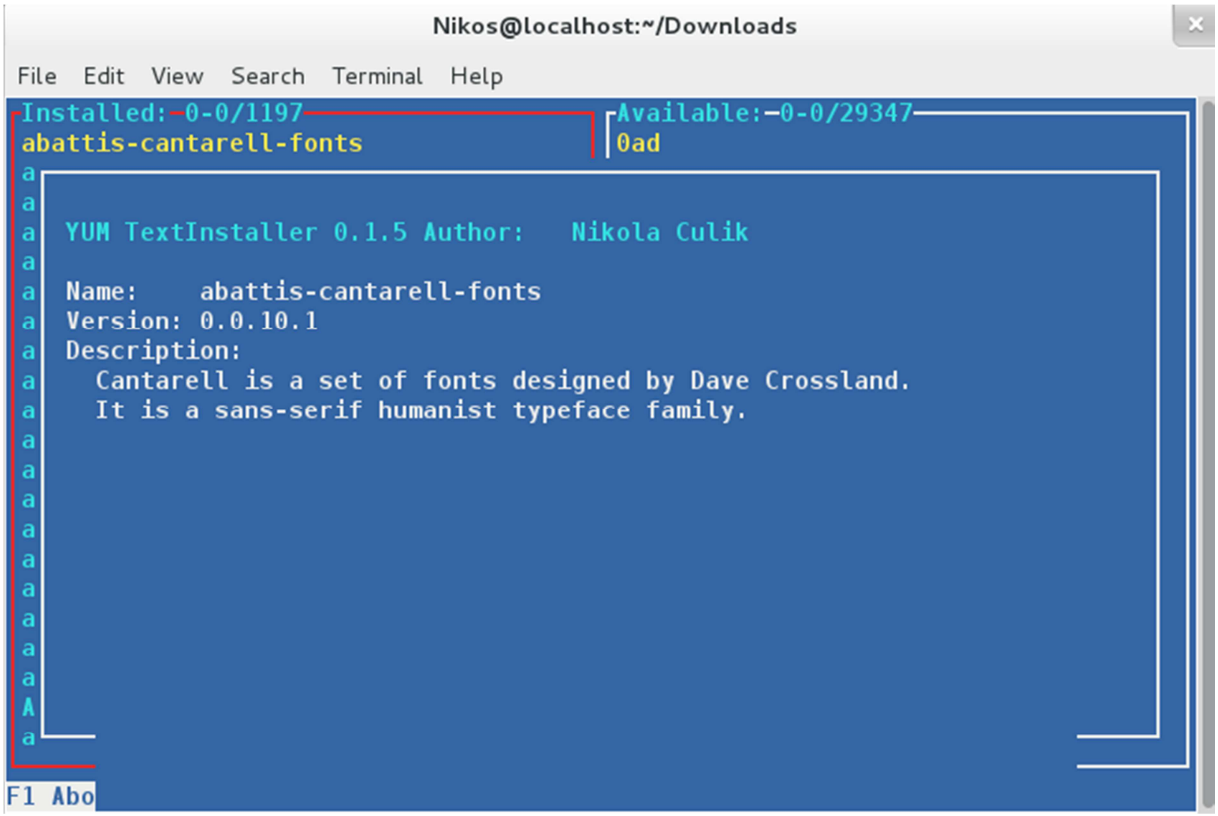
Obrázek 5: Připraven k instalaci



Obrázek 6:Instalace

Během instalace terminál zobrazuje výstup YUMu a rozhraní se obnoví až po dokončení akce.

Balíčky o sobě poskytují velmi mnoho informací jako například Název, verzi, velikost, zaávyslosti nebo krátky popis funkce. Tyto informace jsem uspořádal do speciálního okna které si může uživatel vyvolat pomocí klávesy F11. V současné verzi je uveden pouze název balíčku, jeho verze a krátký popisek, lze však zobrazit i ostatní jmenované parametry zavoláním příslušné funkce. Původní plán umístit tuto funkci na tlačítko F1 jsem opustil protože různá prostředí linuxů ne vždy podporují F1 jako jinou funkci než kontextovou nápovědu právě běžící aplikace, tedy terminálu. Tato změna byla provedena po testování na různých systémech na poslední chvíli a proto není zohledněna v obrázcích této dokumentace.



Obrázek 7:informace o balíčku

Vzhledem k primitivnosti rozhraní je nutné se ujistit že nevyužívá YUM jiný program nebo rozhraní, například čekající aktualizace v jiném uživatelském rozhraní YUM.

Závěr

Zadání práce, vytořit jednoduché textové uživatelské rozhraní pro instalaci a odinstalaci balíčků v Linuxu pomocí balíčkovacího nástroje YUM, bylo splněno. Během práce jsem vytvořil hrubý základ který lze snadno rozšiřovat o další možné funkce nebo modifikovat k vyšší výkonnosti.

Python se ukázal jako velmi dobrý programovací jazyk pro tento úkol neboť mnoho funkcí YUM API počítá právě s použitím pythonu a nebylo tedy nutné provádět složité úpravy nebo převody dat mezi programem a API.

Mezi funkce které jsem nestihl a rozhodně bych je implementoval patří vyhledávání balíčků podle jména a seskupování do kategorií podle využití. Velmi ranné verze programu obsahují i okno na kategorie, bohužel balíčky žádnou takovouto identifikaci neobsahují a bylo by nutné vytvořit seznamy pro každou kategorii, tedy časově náročné.

Seznam použité literatury

[1]*Online Domovské stránky Fedora* [online]. [cit. 2012-03-27]
URL<http://fedoraproject.org>

[2]*Online Domovské stránky YUM* [online]. [cit. 2012-08-15] URL<http://Yum.baseurl.org>

[3] *Distrowatch:Put thefun back unto computing. Use Linux,BSD* [online]. [cit. 2011-01-01]
URL<http://distrowatch.com>

[4] *ncurses 5.9 – GNU Project – Free Software Foundation (FSF)* [online]. [cit. 2011-04-04]
URL<http://www.gnu.org/software/ncurses/ncurses.html>

[5] DICKEY Thomas E, *NCURSES-New Curses* [online]. [cit. 2011-04-04]
URL<http://invisible-island.net/ncurses/>

[6] *Python v2.7.5 documentation* [online]. [cit. 2013-05-12]
URL<http://docs.python.org/2/library/curses.html>

[7] *NCURSES Programming HOWTO* [online]. [cit. 2005-06-20]
URL<http://www.tldp.org/HOWTO/NCURSES-Programming-HOWTO/>

[8] *Dokumentace programu Yumex*
[online].URL<http://www.yumex.dk/p/documnetation.html>

[9] *Filozofie Pythonu*[online]. [cit. 2010-03-16]
URL<http://cs.wikipedia.org/wiki/Filosofie_Pythonu>